

Report 2 Part 2

Module 8: User Interface Design

William H. Smith

INFO-C451

Professor Shawn Dai

December 08, 2022

TABLE OF CONTENTS

Section 1:

Login sequence diagram4

Create a course sequence diagram5

Assign an instructor to a course sequence diagram6

Project Coordination and Progress Report7

Section 2:

Class Diagram.....8

Data Types and Operation Signatures.....9

Traceability Matrix.....10

System Architecture and System Design.....11-12

Global Control Flow.....13

Section 3:

Algorithms and Data Structures.....14

User Interface Design and Implementation.....14-15

Design of Tests.....16-17

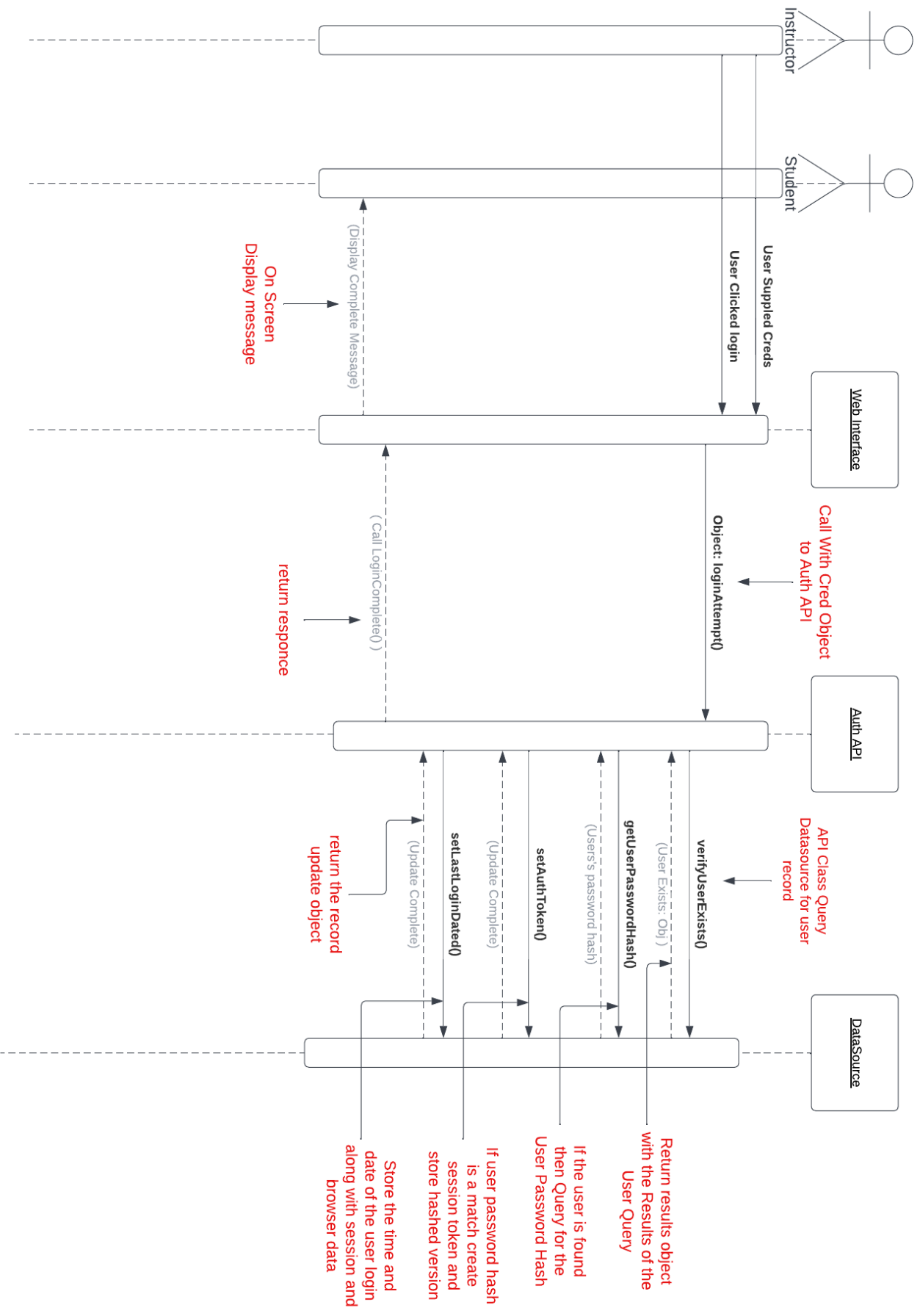
TABLE OF CONTENTS

Project Management:

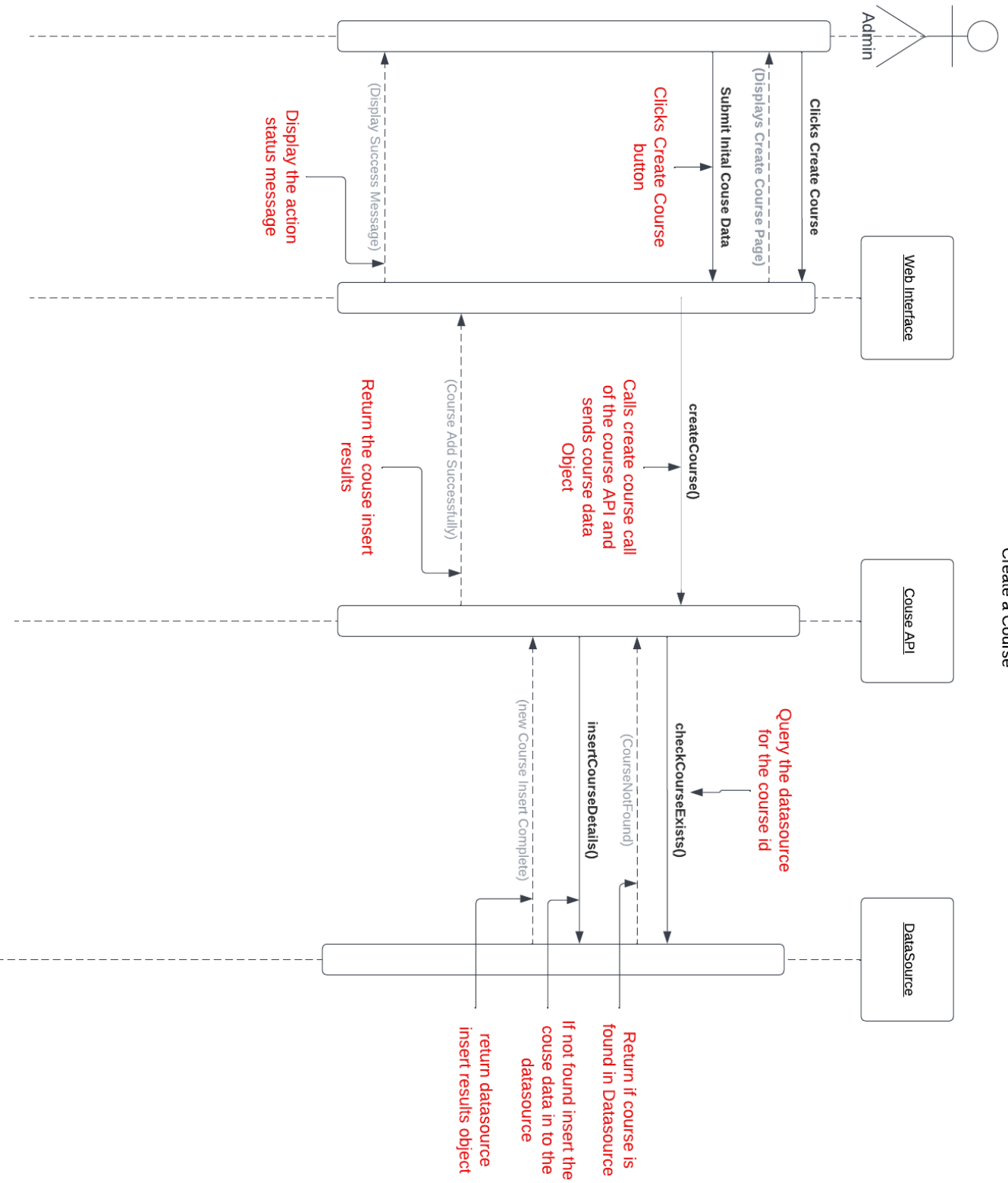
Project Coordination and Progress Report.....18

Plan of Work (Gantt)19-20

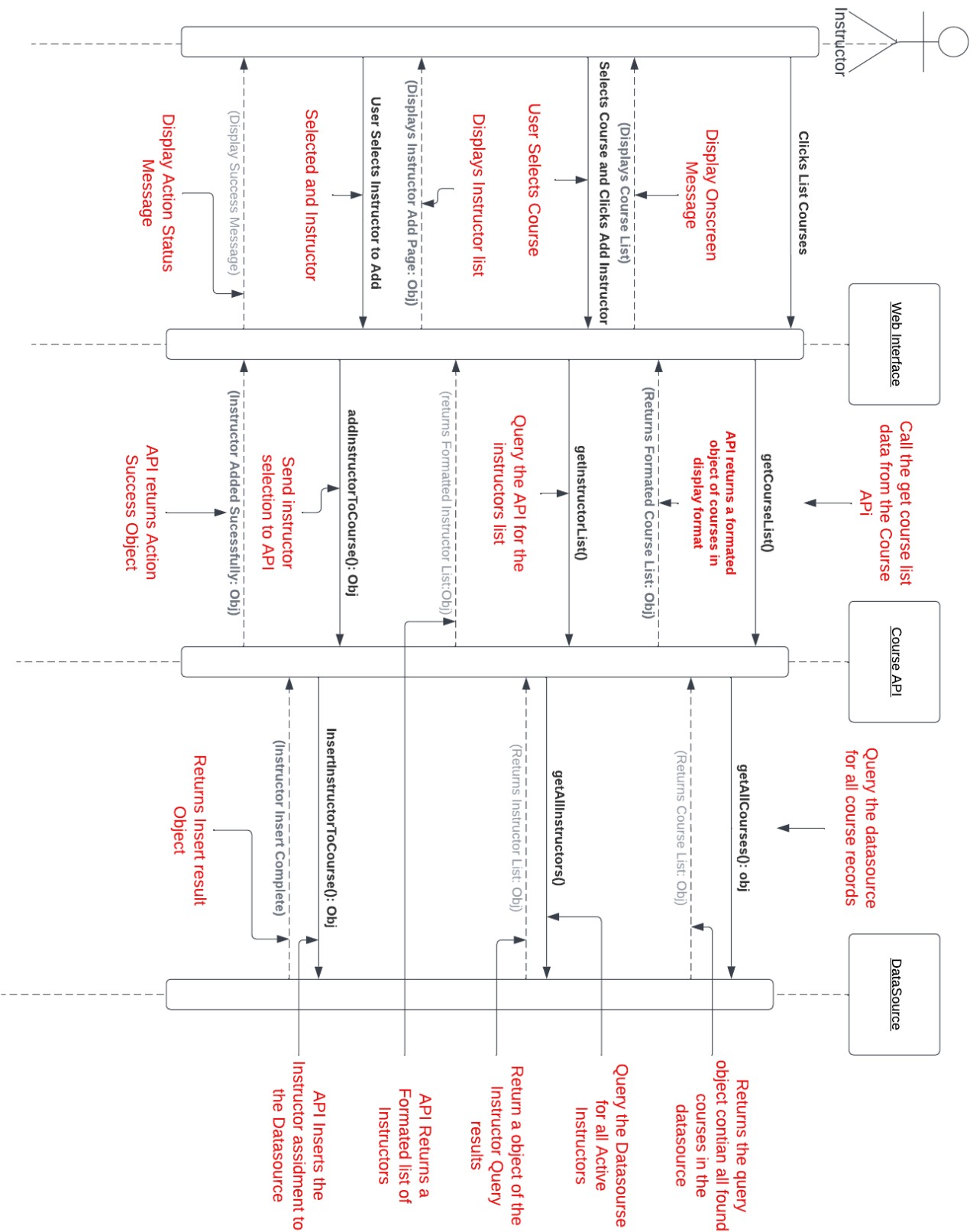
System Login



Create a Course



Add Instructor to Course



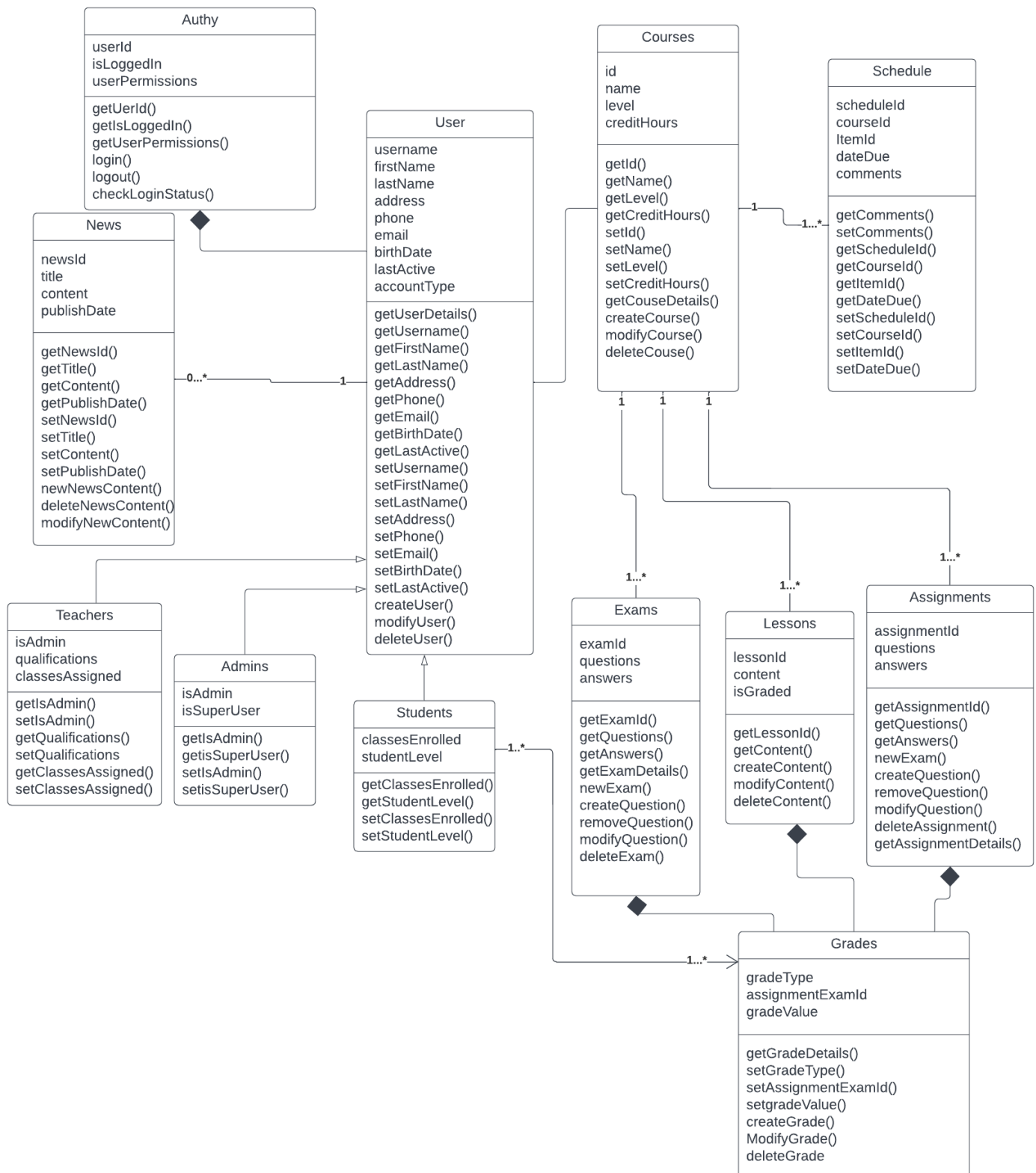
Project Coordination and Progress Report

Currently Implemented Use cases

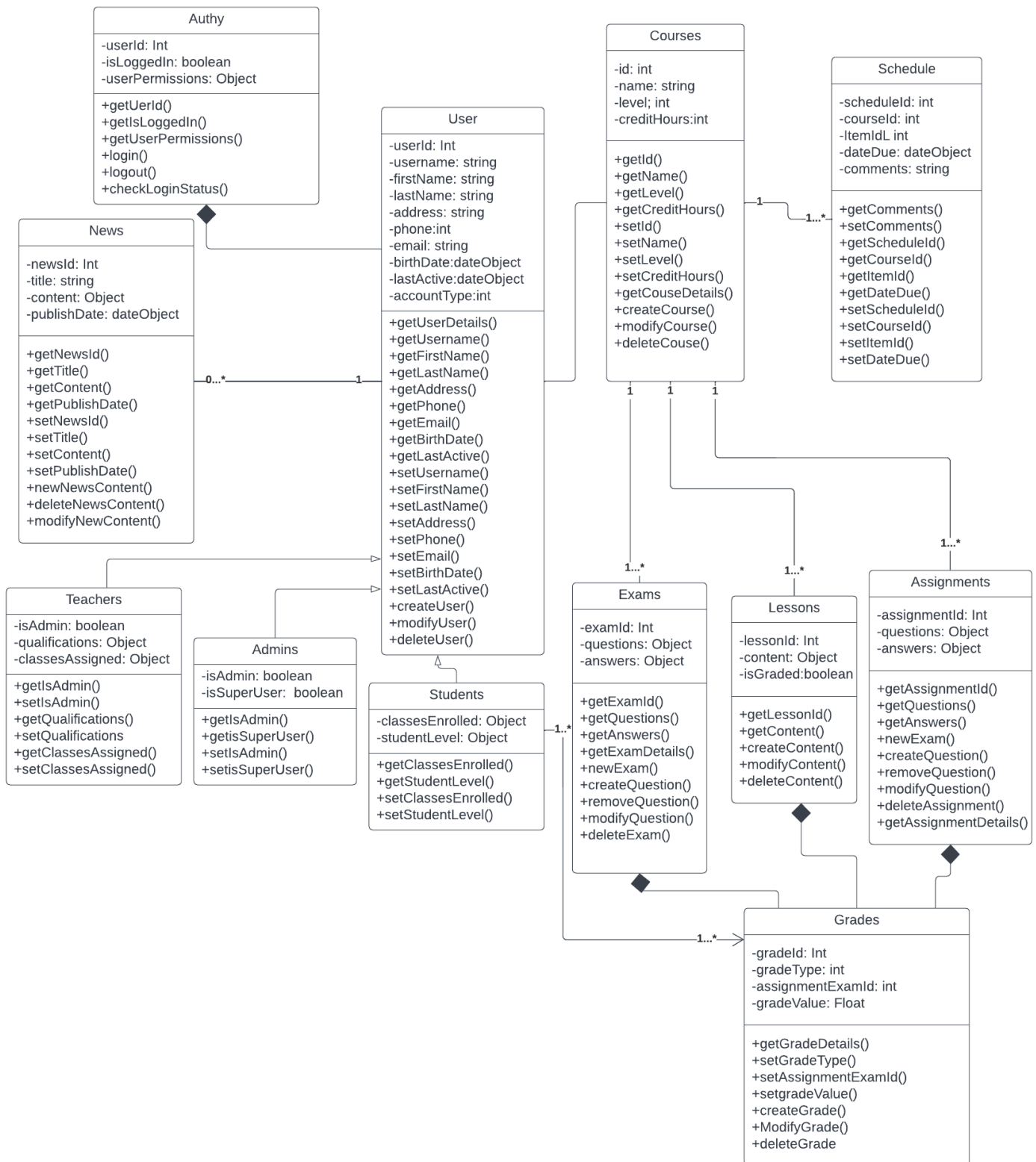
- Password Resets
- Login
- Logout
- View Course List
- View Course Details
- View Course Exams
- View Course Assignments
- View Students List
- View Students Assigned Courses

Currently, the Login process is functional, allowing for proper user sign-in and sign-out. Going along with the Login Module, the password reset is also available. Moving into the course system, you can view the Course list for users and teachers. The detailed content of a course can be viewed, including Assignments, exams, and lessons as well as you can view the students currently assigned to a course as an instructor. Additionally, as an Instructor, you can view a student list for all the courses assigned to an Instructor.

Class Diagram



Data Types and Operation Signatures



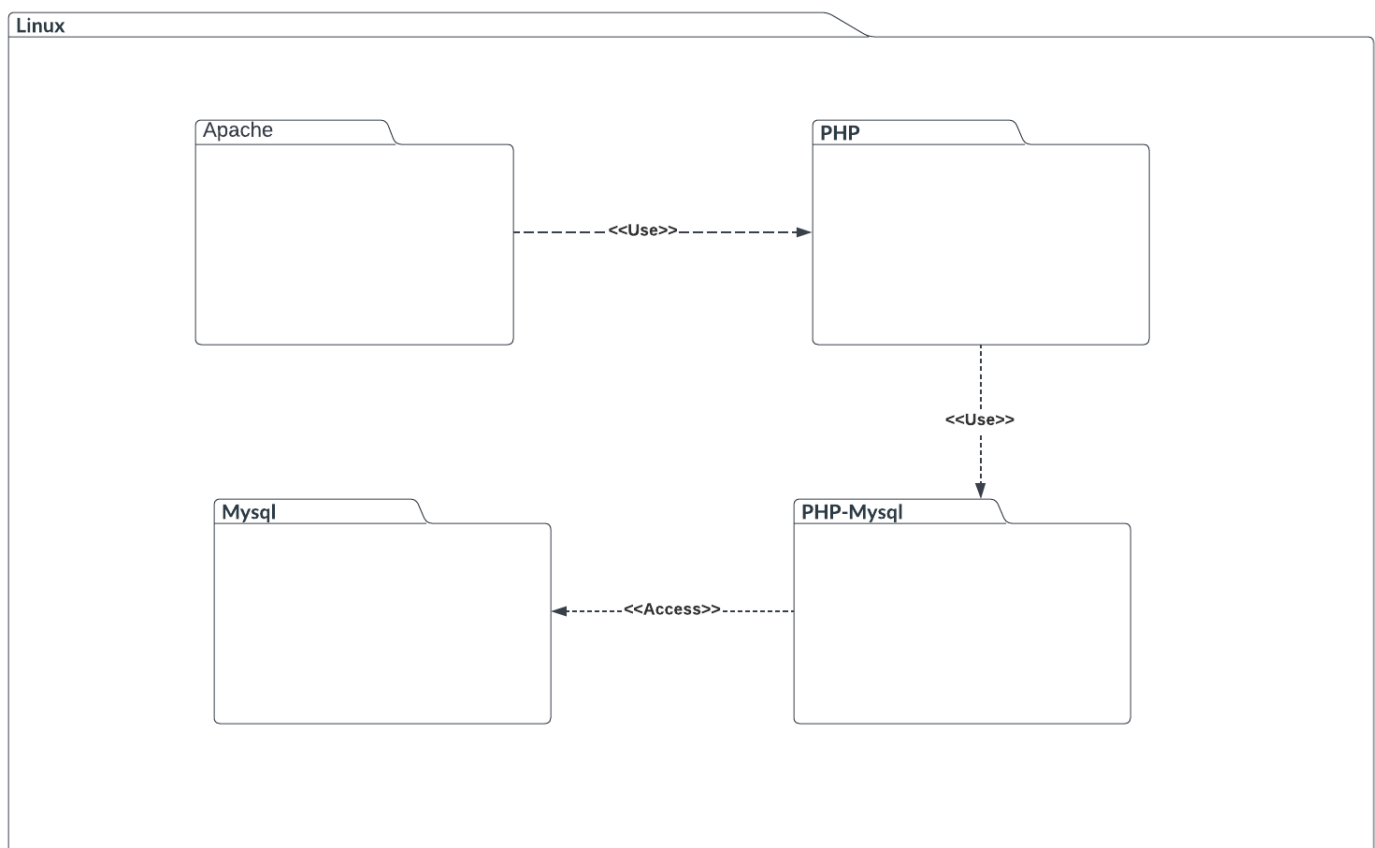
| Requirement name | PW | REQ-1 (Login Screen) | REQ-2 (Scorecard) | REQ-3 (Splash) | REQ-4 (Schedule) | REQ-5 (Course Management) | REQ-6 (News Bc) | REQ-7 (Profile) | REQ-8 (Acct. Settings) | REQ-9 (Resource) |
|---------------------------------------|----|----------------------|-------------------|----------------|------------------|---------------------------|-----------------|-----------------|------------------------|------------------|
| Login () | 5 | X | | | | | | | | |
| Logout | 5 | X | | | | | | | | |
| Change User Password | 2 | X | | | | | | | | |
| View Course Calendar | 3 | | | | X | | | | | X |
| View Course List | 4 | | | X | | | X | | | X |
| View Course Details | 4 | | | | | | | X | | X |
| View Course Exams | 4 | | | | | | | | | X |
| View Course Assignments | 4 | | | | | | | | | X |
| View Course Lessons | 4 | | | | | X | | | | X |
| Submit Course Assignment | 4 | | | | | | | | | X |
| Submit Course Exam | 4 | | | | | | | | | X |
| Submit Grades | 4 | | X | | | | | | | X |
| View Student List | 3 | | | | | X | | | | |
| View Student Profile | 1 | | | | | | | X | | X |
| View Students Assigned Courses | 4 | | | | | X | | | | |
| View Student Course Grades | 3 | | X | | | X | | | | |
| View Instructor's List | 2 | | | | | X | | | | |
| View Instructor Profile | 1 | | | | | | | X | | X |
| View Assigned Instructor Courses | 3 | | | | | X | | | | |
| Add/Edit Course | 3 | | | | | X | | | | |
| Add/Edit Student | 4 | | | | | X | | | | |
| Add/Edit Instructors | 3 | | | | | X | | | | |
| Add/Edit Admin | 3 | | | | | X | | | | |
| Add/Remove Student to/from Course | 3 | | | | | X | | | | |
| Add/Remove Instructor to/from Course | 3 | | | | | X | | | | |
| Add/Remove Assignment to/from Course | 3 | | | | | X | | | | |
| Add/Remove Calendar Item For a Course | 3 | | | | | X | | | | |
| Add/Remove Exam to/from Course | 3 | | | | | X | | | | |
| Add/Remove Lesson to/from Course | 3 | | | | | X | | | | |
| Max PW | 5 | | 4 | 4 | 3 | 4 | 4 | 4 | 3 | 4 |
| Total PW | 12 | | 7 | 7 | 3 | 66 | 4 | 6 | 4 | 32 |

System Architecture and System Design

- Architectural Styles

With this mainly being composed as a web-based application, The main architectural style used in the building of the application is Three-tier Patten. This approach manifests as this application is a web-based front-end. We are also utilizing a PHP backend for all the API calls to access the system data objects and a SQL database to store all content and user information.

- Identifying Subsystems



- Mapping Subsystems to Hardware

- Will your system run on multiple computers? For example, whether you will have a client (web browser) and a server (webserver)

Yes, the system will run on multiple systems, utilizing a webserver "Apache" to host the application's front end. This allows for access from any computer with network access to the system.

- Persistent Data Storage
 - Whether your system requires to save data outliving a single execution of the system?

Yes, The system will utilize storage outside of the single execution scenario. We will use a SQL database for persistent storage, storing all user data, course data, grades, student, instructor, and system setting information to be stored persistently through system restarts.

- Network Protocol

The system will utilize the HTTPS/HTTP network Protocol to allow access to the UI from any Computer with access to the Systems Network. I choose this protocol because it has the most out-of-the-box Compatibility with just about every device on the market over the last five years, allowing for easy adoption of the application into almost any environment.

Global Global Control FlowControl

- Execution order: Is your system procedure-driven and execute in a linear fashion, where every user every time has to go through the same procedures or is it an event-driven system that waits in a loop for events, and every user can generate actions in a different order?

The system is event-driven. A user must click a button or enter some text for the system to proceed. Otherwise is in a loop waiting for user input to execute other processes.

- Time dependency: Do you have timers for your system? Is your system of event-response type, with no concern for real-time (periodic), or is it a real-time system? If it is periodic, what are the time constraints for each period?

The system is an Event response type, with a response limit of 30 seconds for most handler tasks.

- Concurrency: Does your system use multiple threads?

Not directly, but the Webserver Apache starts a new thread with every connection to the web server from each client.

- Hardware Requirements

Hardware Requirements for the system are any X86, or Arm64-based system running the Apache and PHP Packages. You will need at least 1 GB of Ram and 20 GB of Hard drive space. Additionally, you will need at least 10 Mbps of Networking throughput, but 1 Gbps is recommended for small deployments

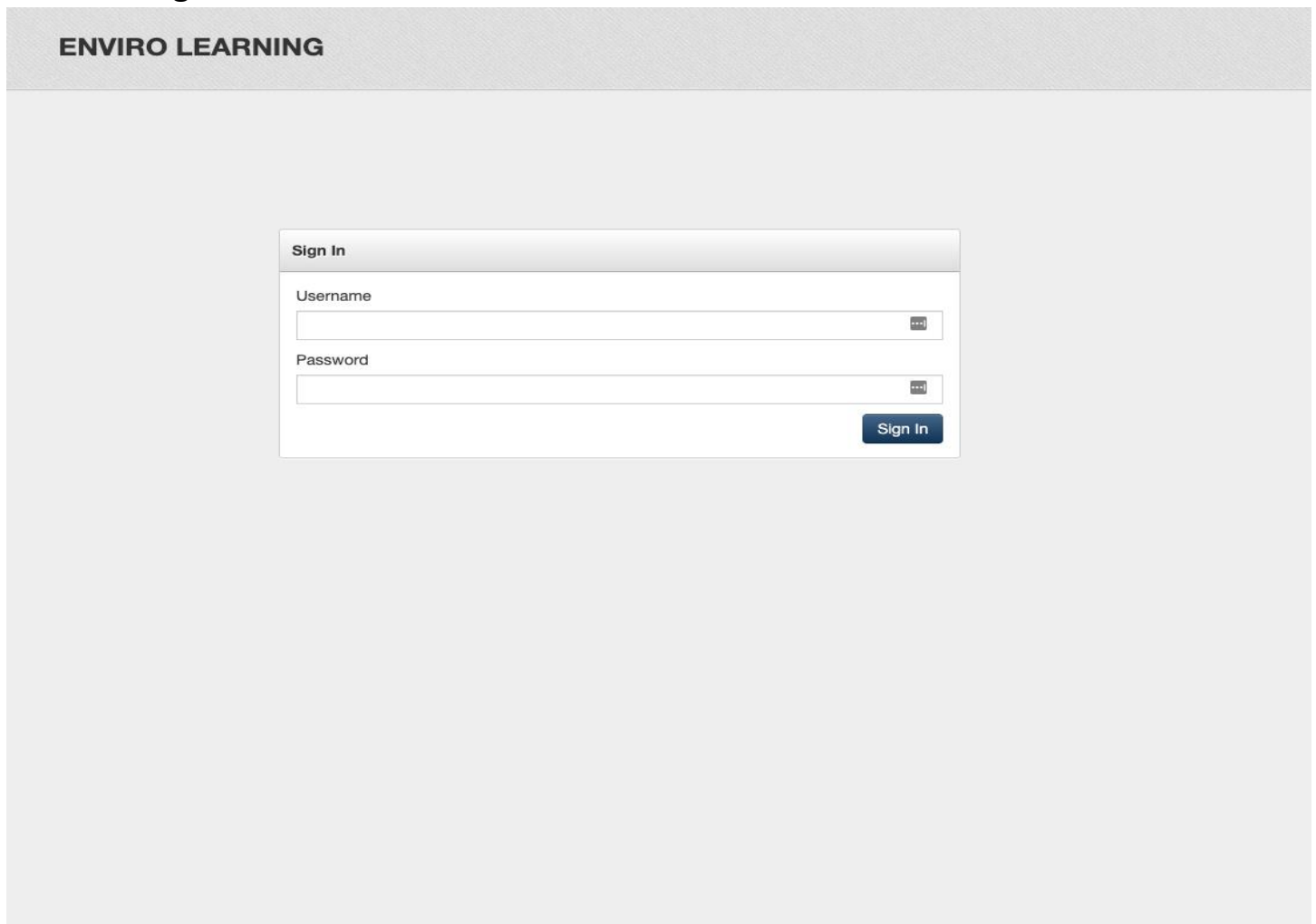
Data Structure

Currently in the design of the platform it utilizes arrays for the temporary storage on complex data related to the Details of User, Courses, Assignment, Exams, and the Grads that are the result of these assignments and Exams. The choice to utilize that Array data structure was chosen because of its more native compatibility with the Server side programming language PHP which is utilized for all server side processing tasks responsible for storing and retrieving data for use with Web based applications.

User Interface Design and Implementation.

The user interface has changes in only a few ways since the implementation of the Design mockups. The layout of the initial mockups was I have continued to develop the UI the simple flow of high level categories to the low level or more granular function as an example you flow through the UI process as an administrator from users to classes to assignments or Exams and then grades. Process flows always follow from highest or most broad data to most specific data for a given category.

Current Login Screen



ENVIRO LEARNING

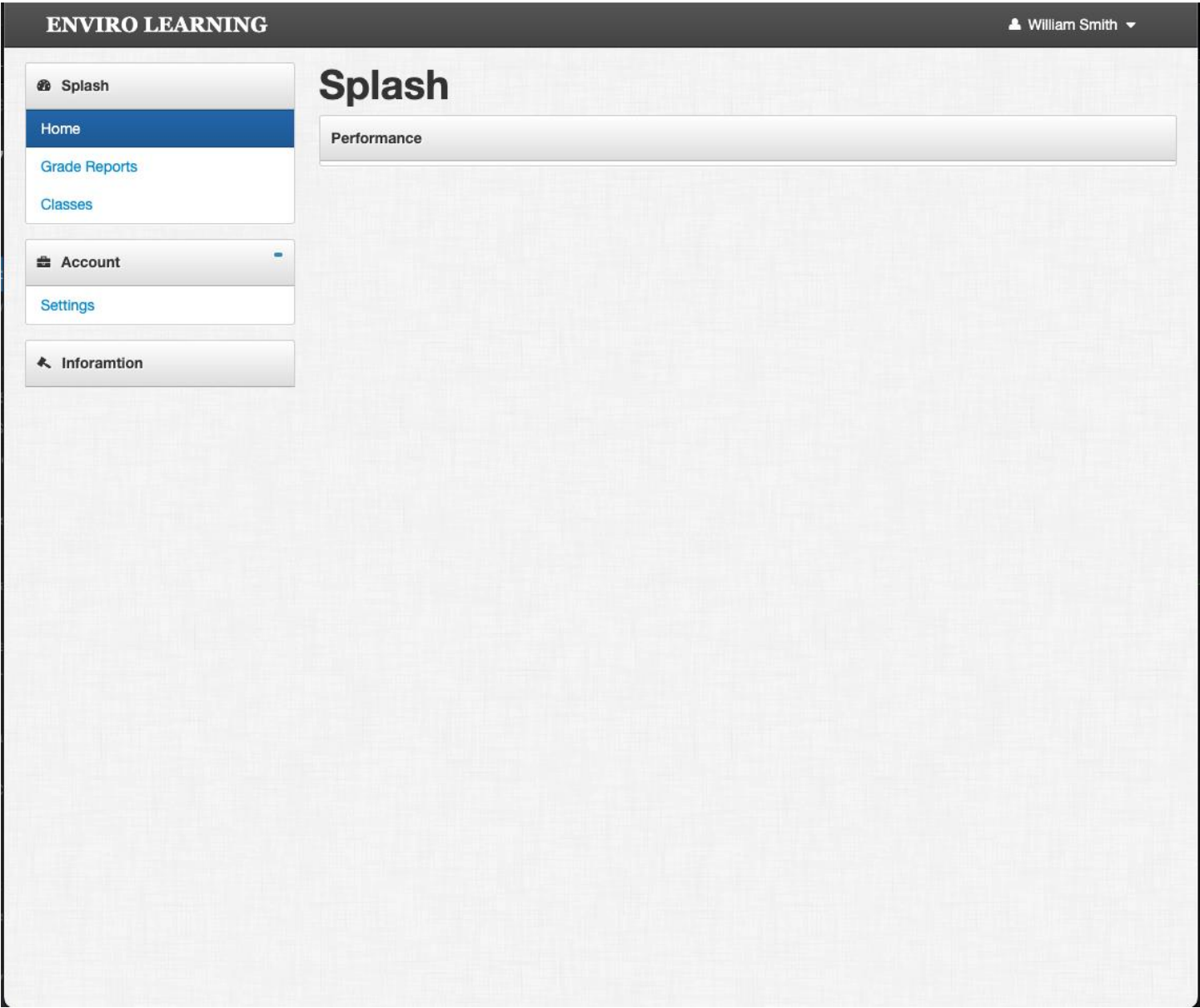
Sign In

Username

Password

Sign In

Current Dashboard Layout



Design of Tests

While conducting the unit testing, the PHP-based calls will Provide and process the data and information changes and verify different types of information through the numerous functions of the VLE system. I will be utilizing the PHPUnit testing framework to write and verify the functions and operations of the system. The system will be written into a simple execution and PHP file with individual testing classes for each function, demonstrating the proper testing of each call from executing a simple PHP file via the web or PHP-CLI call to the file.

1. Login:

The login unit test will start by calling the “common.php” with the switch parameter of login verify. Additionally, this will provide a set of testing credentials in the post parameter of username and password and, with check, the return status for an error state, which reasserts a failure as well as a timeout, or in the event that the credentials come back as invalid all will return a failure state for the Unit test. The passing result is if the call returns a valid response resulting in valid credentials returning a pass status for the test.

2. Change User Password:

The change password unit test will start by calling “common.php” with the switch parameter of the change password. This will only allow changing of a password while a user is logged in. we will attempt to assert a change password without a proper login first and expect a failure as a result. Assuming a passing result or a failure, we send proper test credentials and check from the proper login. If proper login is verified, we attempt to change the login password if a unit test fails. If successful, we return a unit test passed.

3. View Course Calendar:

The View Course List Unit test will call the view Course list function from the course class to check if the results come back with an array of courses greater than zero, then check the array data for a few hard-coded vatable to verify they contain information. Suppose they return greater than zero in the array result, and the array contains data based on asserted information. In that case, we return the unit test pass otherwise, fail is returned for any other results.

4. Submit Course Assignment:

The submit Course Assignment unit test will call the submit Assignment function of the assignments class and pass sample assignment details to the variables. After it, it will wait and check the response for a successful status and then query the database to verify the accuracy of the test information in the system.

5. View Student List:

The unit test for View all available students will call the get students function of the student class. This will wait until a response is received or the timeout has been reached. If the request returns with a result, it will check the returning result for proper data integrity against a known hash. If the data returned is valid, then we return a unit test passed if the request does not return before the timeout or with invalid data, then we return a unit test fail.

6. View Students Assigned Courses:

The unit test for viewing assigned courses will make a call to the "getAssignedCourses: of the student class with the student_id parameters filled with a testing value for matching sample data. We will wait until a response is returned or the timeout limit is reached. If the call returns with data, then we check the return data against a known data hash. If the data is valid, then we return a true result. If the data is invalid on the integrity check, then we return a test failure. Additionally, if no result is returned before the timeout, then we return a unit test failure.

7. Add/Edit Course;

The unit test for creating a new course will start by calling the "newCourse" function of the courses class with the function parameters completed with available testing data. The test will wait for a status response or until the timeout threshold has been reached. If the result returns with a successful status, we will manually access the database and check the data integrity of the entry. If the known data matches the data in the entry, then we return a unit test passed. Otherwise, we return the unit test failed. Additionally, if there is no response from the function call before the timeout occurs, we will return a unit test failure as a default.

8. Add/Remove Student to/from Course:

The unit test for adding a student to a course is started by calling "addStudent" function from the courses class. The function is provided with the test "student_id" as well as the testing "course_id" if the function call returns before the timeout has been reached, then we manually check if the student has been added to the database as a member of the course. If we find the student, we return a pass for the unit test. If the student is not found, then the test will fail. Additionally, the test will fail by default if the timeout is reached before the function returns.

Project Coordination and Progress Report

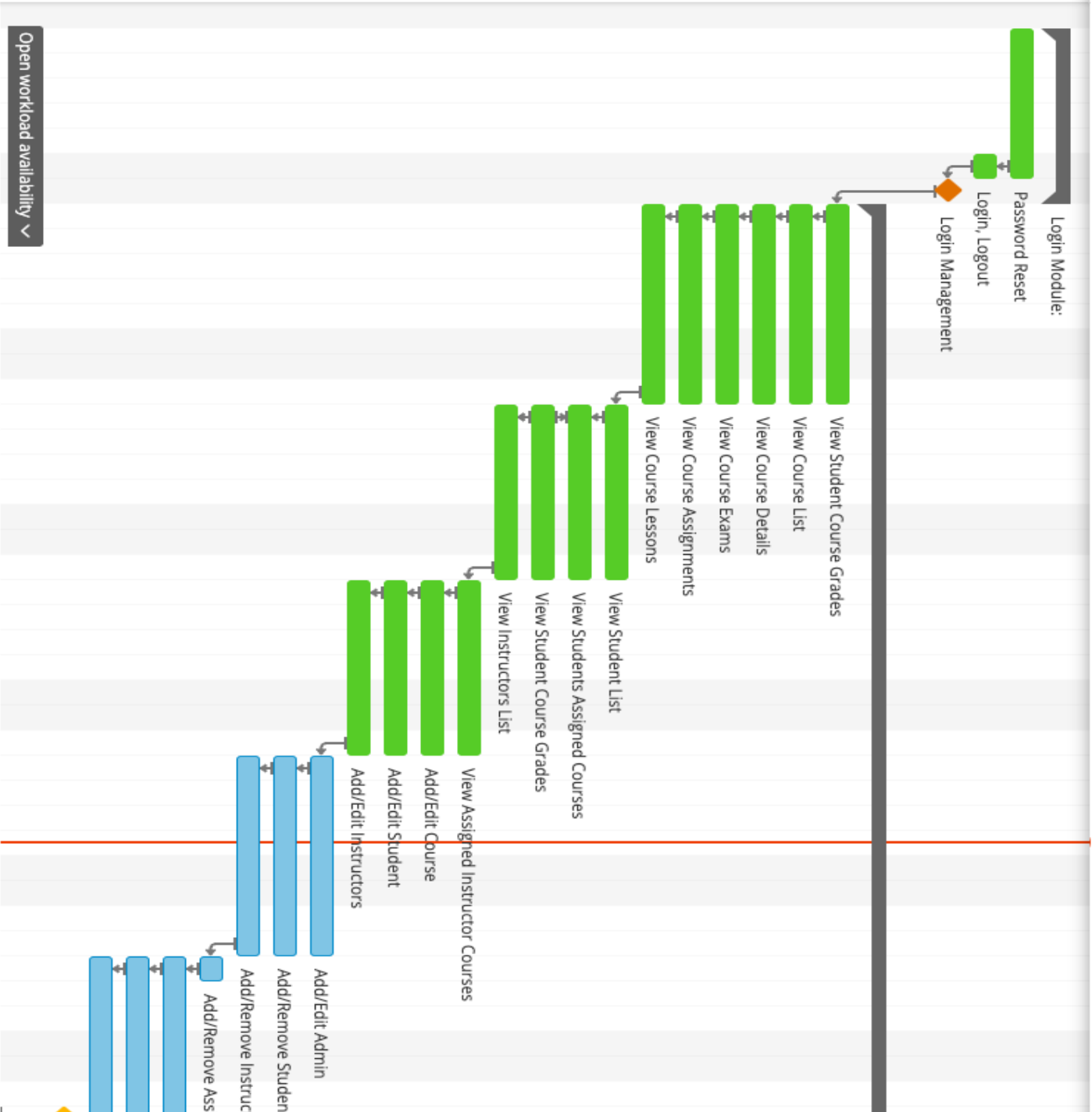
The use cases that are currently implemented

- Login / Logout
- View Course List
- View Course Details
- View Course Exams
- View Course Assignments
- View Course Lessons
- View Students List
- View Students Assigned Courses
- View Students Grades
- View Instructors List

Complete and Currently in Progress

What Currently Functional is most of the View Methods for most of the data needed to be displayed in the system. What currently being worked on is the Functions and data structures that are required to submit that data with data integrity verification and additional checks required for consistent data entry.

| Login Module: | | | 100% |
|--------------------------|---|--------------------------------|------|
| 1 | ✓ | Password Reset | 100% |
| 2 | ✓ | Login, Logout | 100% |
| 3 | ✓ | Login Management | 100% |
| + Add task + Add section | | | |
| Course Management: | | | 68% |
| 6 | ✓ | View Student Course Grades | 100% |
| 7 | ✓ | View Course List | 100% |
| 8 | ✓ | View Course Details | 100% |
| 9 | ✓ | View Course Exams | 100% |
| 10 | ✓ | View Course Assignments | 100% |
| 11 | ✓ | View Course Lessons | 100% |
| 12 | ✓ | View Student List | 100% |
| 13 | ✓ | View Students Assigned Co... | 100% |
| 14 | ✓ | View Student Course Grades | 100% |
| 15 | ✓ | View Instructors List | 100% |
| 16 | ✓ | View Assigned Instructor Co... | 100% |
| 17 | ✓ | Add/Edit Course | 100% |
| 18 | ✓ | Add/Edit Student | 100% |
| 19 | ✓ | Add/Edit Instructors | 100% |
| 20 | ✓ | Add/Edit Admin | 0% |
| 21 | ✓ | Add/Remove Student to/fro... | 0% |
| 22 | ✓ | Add/Remove Instructor to/f... | 0% |
| 23 | ✓ | Add/Remove Assignment to... | 0% |
| 24 | ✓ | Add/Remove Calendar Item... | 0% |
| 25 | ✓ | Add/Remove Exam to/from ... | 0% |
| 26 | ✓ | Add/Remove Lesson to/fro... | 0% |
| 27 | ✓ | Course Management | 0% |



451

Read-only view, generated on 15 Oct 2022

